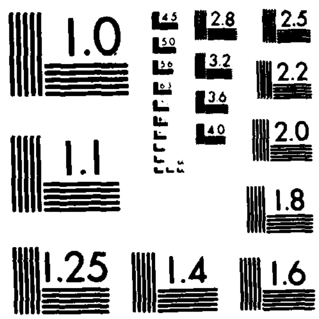
AD-A129 549 SOME COMMON-SENSE OPTIMIZATION TECHNIQUES FOR NON-DIFFERENTIABLE FUNCTION..(U) NORTH CAROLINA AGRICULTURAL AND TECHNICAL STATE UNIV GREENSBO.. 1/1

UNCLASSIFIED B N BORAH ET AL. 02 JUN 83 ARO-17435.1-MA-H F/G 12/1 NL

END
DATE
FILMED
7 83
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

Some Common-Sense Optimization Techniques for Non-Differentiable Functions of Several Variables

By

B.N. Borah

and

J.F. Chew

Research sponsored by
U.S. Army Research Office,
Research Triangle Park, N.C.

SOME COMMON-SENSE OPTIMIZATION TECHNIQUES FOR NON-DIFFERENTIABLE FUNCTIONS OF SEVERAL VARIABLES

Dr. Bolindra N. Borah, Professor of Applied Mathematics and Computer Science

Dr. James F. Chew, Associate Professor of Mathematics

June 2, 1983

U. S. Army Research Office

DAAG29-80-G-0004

North Carolina Agricultural and Technical State University

Approved for Public Release;
Distribution Unlimited

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER FINAL REPORT	2. GOVT ACCESSION NO. AD-A129549	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SOME COMMON-SENSE OPTIMIZATION TECHNIQUES FOR NON DIFFERENTIABLE FUNCTIONS OF SEVERAL VARIABLES		5. TYPE OF REPORT & PERIOD COVERED 6/1/80 - 6/30/83 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dr. Bolindra N. Borah Dr. James F. Chew		8. CONTRACT OR GRANT NUMBER(s) DAAG29-80-G-0004
9. PERFORMING ORGANIZATION NAME AND ADDRESS Mathematics & Computer Science Dept. North Carolina A&T State University Greensboro, N. C. 27411		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office Post Office Box 12211 Research Triangle Park, NC 27709		12. REPORT DATE June 2, 1983 13. NUMBER OF PAGES 43
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Resident Representative Georgia Institute of Technology Room 325, Hinman Research Building Atlanta, GA. 30332		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Methods of Global Optimization for Non-Differentiable Functions

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SEE ATTACHED SHEETS

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SOME COMMON-SENSE OPTIMIZATION TECHNIQUES FOR NON-DIFFERENTIABLE FUNCTIONS OF SEVERAL VARIABLES

ABSTRACT:

The problem of obtaining global optima of non-differentiable functions of several variables is studied. In general, the functions are multimodal and continuous on a compact domain. Two distinct methods are proposed and to some extent compared: The method of systematic search and the random search technique. The method of uniform saturation [the one variable version of the systematic search method] is based on bisecting the interval (in the one-variable case) repeatedly. Without loss of generality, we may restrict the discussion to the closed unit interval $I = [0,1]$. At the first stage, $n = 1$, bisect the interval I using the point $x = 1/2$. Let $M_1 = \max [f(1/2), f(1)]$. At the second stage, $n = 2$, bisect each of the intervals $[0,1/2]$ and $[1/2,1]$ using the points $x = 1/4$ and $x = 3/4$ respectively. Let $M_2 = [m_1, f(1/4), f(3/4)]$. By the nth stage we would have subdivided the interval I into 2^n subintervals, each of length $(1/2)^n$, wherein the partition points over and above those previous stages are $i(1/2)^n$, $i = 1,3,\ldots,2^n-1$. Thus the M_n's are inductively given by $M_n = \max[M_{n-1}, f(i/2^n); i = 1,3,\ldots 2^n-1]$. It is now clear that M_n is monotonic increasing sequence, $M_1 \leq M_2 \leq M_2 \leq \ldots$ If we repeat the procedure enough times, we would "saturate" the interval I by evenly spaced points in such a way that the distance between

two neighboring points diminishes geometrically as n-increases. Thus we "zero-in" on a solution of the problem. This method is later modified to the case of functions of two or three variables.

The Random Search Technique used here determines all the optimal points of the non-differentiable continuous functions with many variables defined on compact domain. The procedure begins with evaluating the given function at pre-determined number of points selected randomly over the closed bounded domain. Suppose m points are selected randomly over the domain and the function is evaluated at each of the m points. The minimum functional value and the point at which the minimum occurs (if the problem is one of minimization) are saved. This step is carried out n times, where n is sufficiently large. The resulting n points will cluster around the minima. Suppose there are r cluster points, then there is a possibility that around each cluster point, a local minimum may exist. We develop a single program to find all the cluster groups as well as cluster points using a local optimization routine. Thus the global minimum is obtained by simple comparison. The new method developed here is clearly an improvement with regards to time and accuracy over the methods proposed by Becker and Lago and Price's CRS procedure.

TABLE OF CONTENTS

We would like to thank the U. S. Army Research Office for providing us funds to do this research. Also we must thank Ms. Sitrena McLendon for helping us in preparing and typing this technical report.

INTRODUCTION

There are many optimization procedures which enable one to determine the minimum of a unimodal function in n-space. If the function is differentiable in a compact domain, global minima may be obtained through the use of derivatives. However, the problem of global optimization of multimodal function has received comparatively little attention, more so when the function in question is non-differentiable. No efficient method has been developed to tackle global optimization problems.

As a general principle, the accuracy with which a procedure locates optima improves with the number of functional evaluations. In principle, however, one seeks a balance between a degree of certainty and the cost of implementation. A procedure which locates optima with great precision and certainty would be practically worthless if it requires economically unfeasible number of calculations.

There are several methods presently utilized to seek global optimua; among them are those suggested by Brooks [1], Becker and Lago [2] and Price's CRS method [3]. The Simple Random Method accepts the optimum function value as global optimum after making a specified number of trials randomly selected from the domain. The stratified Random Search method divides the domain into a number of subdomains of equal size and selects, at random, a trial point from each subdomain and each time keeps the optimal function value. The procedure is repeated a good many times. Some improvement on the simple random search is provided by Becker and Lago. Their

procedure begins with a Simple Random Search over the domain, instead of retaining the single point with the optimal function value, Becker and Lago retain a predetermined number of points with optimal function values in each trial. If the number of trials is sufficiently high, the retained points tend to cluster around some optima. Then a mode seeking algorithm is used to group the points into discrete clusters and to define the boundaries of the subregions each embracing a cluster. The clusters are graded, by searching in each for the retained points with the lowest function value and then rated according to the relative values of the cluster minima. The entire procedure is then repeated using as the initial search region that subdomain, defined by the mode seeking algorithm around the 'best' cluster. The user may choose to examine also the second best cluster, or indeed all clusters, according to the extent of his doubt as to whether or not the global minimum will be found in the subdomain defined by the best cluster.

The controlled Random Search (CRS) suggested by Price is similar to Becker and Lago, but CRS combines the random search and mode-seeking algorithm into a single continuous process. But the problems of inefficiency and economic consideration still remain.

METHOD OF SOLUTIONS

This paper deals with two methods: (I) Systematic Search (The Method of Uniform Saturation), (II) Random Search. In both cases it is assumed that the functions are defined and continuous on a compact domain. They are also assumed to be multimodal functions. In general the systematic search does not provide all the optimal points, the primary emphasis here being location of a global optimum.

Despite several restrictions and difficulties, the Random Search method attempts to obtain all the optima, one optimum point in each mode.

I. SYSTEMATIC SEARCH (The Case of Two or Three Variables)

Suppose $f(x,y)$ is continuous on the closed unit square $S = \{(x,y): 0 \leq x,y \leq 1\}$. Then certainly a subdivision of the interval $[0,1]$ into, say, 50 equal subintervals would have to be considered as a reasonable partition. That is to say, 50 is a reasonably small number. Yet even with 50 partition points on each of the x and y axes, we are faced with 50 x 50 = 2500 partition points of the unit square S. For the case of a function $f(x)$ of one variable, we certainly would want to partition the interval $[0,1]$ into MORE than 50 subintervals to get a reasonable assurance that an optimum has been included. Hence we cannot be confident that the global optimum will be among the values at the 2500 partition points of the square S.

The case of a function of three variables is much worse. Here a subdivision of each co-ordinate AXIS into 50 partition points results in 50 x 50 x 50 = 125,000 points of the cube. This is just to get a crude starting point. Hence we see that the number of evaluations becomes prohibitive very rapidly and so, to have any hope whatsoever of handling the multivariable case, we would have to abandon the purely exhaustive scheme (Method of Uniform Saturation) used in the univariable case.

The proposed method is based on two steps. The first step involves consideration of an initial grid on the domain. An initial

point is then obtained based on the grid. The second step starts with the initial point and proceeds by the method of 'crossings'.

Any direct search procedure such as the one presently given would require a large number of evaluations. For a function $f(x,y)$ of two variables on a rectangle, we consider 100 partition points on each of the x and y axes to be reasonable. This gives rise to $100 \times 100 = 10{,}000$ partition points. We realize that 10,000 evaluations might not be cost-effective and that other more efficient methods might be employable. The fact remains that this procedure is direct, simple to execute and self-contained (not based on other search procedures already in existence).

Several theorems pertaining to functions of two variables are proved and some twenty one illustrative, computational examples are provided. These examples comprise Tables 1, 2 and 3. The computer programs are given in Appendix A.

The One-Variable Case: (The Method of Uniform Saturation)

Consider the non-linear programming (NLP) problem: MAXIMIZE $f(x) : a \leq x \leq b$, where $f : I \to R$ is a continuous real-valued function defined on the closed interval $I = [a,b]$. Without loss of generality, we may restrict the discussion to the closed unit interval $I = [0,1]$. At the first stage, $n = 1$, bisect the interval I using the point $x = 1/2$. Let $M_1 = \max\{f(1/2), f(1)\}$. At the second stage, $n = 2$, bisect each of the intervals $[0,1/2]$ and $[1/2,1]$ using the points $x = 1/4$ and $x = 3/4$ respectively. Let $M_2 = \max M_1, f(1/4), f(3/4)$. At the third stage, $n = 3$, bisect each of the intervals $[0,1/4], [1/4,1/2], [1/2,3/4]$, and $[3/4,1]$ using the points $x = 1/8$, $x = 3/8$, $x = 5/8$, and $x = 7/8$ respectively. Set $M_3 = \max\{M_2, f(i/8) : i = 1,3,5,7\}$.

By the n'th stage we would have subdivided the interval I into 2^n subintervals, each of length $(1/2)^n$, wherein the new partition points over and above those of the previous stages are $i(1/2)^n : i = 1,3,\ldots,2^n - 1$. Thus the M_n's are inductively given by $M_n = \max\{M_{n-1}, f(i/2^n) : i = 1,3,\ldots,2^n - 1\}$. It is now clear that M_n is monotone increasing, viz. $M_1 \leq M_2 \leq M_3 \ldots$ If we repeat the procedure enough times, we would "saturate" the interval I by evenly spaced points in such a way that the distance between two neighboring points diminishes geometrically as n increases. Thus we "zero in" on a solution of the problem. That is, if x_o SOLVES the problem, then there is a bisecting point x_k WITHIN ANY PRESCRIBED DISTANCE from x_o. Thus if $\varepsilon > 0$ is preassigned, we are assured of the existence of an x_k for which $|x_k - x_o| < \varepsilon$ whenever

n is such that $2^n > 1/\varepsilon$. Since the function f(x) is continuous, we know that $f(x_k)$ will be close to $f(x_o)$ whenever x_k is "sufficiently" close to x_o.

The Two-Variable Case

We next consider a real-valued function f(x,y) which is continuous on the closed unit square $S = \{(x,y): 0 \leq x,y \leq 1\}$. The non-linear programming (NLP) problem is: MAXIMIZE $f(x,y) : (x,y) \varepsilon S$.

Theorem 1 Let f(x,y) be a real-valued function which is continuous on a compact domain D. Then

$$\underset{(x,y)\varepsilon D}{\text{MAXIMUM}}\ f(x,y) = \underset{x\varepsilon D_y}{\text{MAX}} \{\underset{y\varepsilon D_x}{\text{MAX}}\ f(x,y)\},$$

where for each fixed x, $D_x = \{y : (x,y) \varepsilon D\}$ and for each fixed y, $D_y = x:\{(x,y) \varepsilon D\}$.

Proof Clearly $\underset{(x,y)\varepsilon D}{\text{MAXIMUM}}\ f(x,y) \geq \underset{x\varepsilon D_y}{\text{MAX}} \{\underset{y\varepsilon D_x}{\text{MAX}}\ f(x,y)\}$. Suppose that the inequality is strict:

$\underset{(x,y)\varepsilon D}{\text{MAXIMUM}}\ f(x,y) > \underset{x\varepsilon D_y}{\text{MAX}} \{\underset{y\varepsilon D_x}{\text{MAX}}\ f(x,y)\}$. Say

$\text{MAXIMUM}\ f(x,y) = f(x_o, y_o)$

then

$$f(x_o, y_o) > \underset{x\varepsilon D_y}{\text{MAX}} \{\underset{y\varepsilon D_x}{\text{MAX}}\ f(x,y)\} \geq \underset{y\varepsilon D_x}{\text{MAX}}\ f(x_o, y) \geq f(x_o, y_o),$$

a contradiction.

As a corollary, we have:

If f(x,y) is continuous on the closed unit square S, then

$$\underset{(x,y)\varepsilon S}{\text{MAXIMUM}}\ f(x,y) = \underset{0\le x\le 1}{\text{MAX}}\ \{ \underset{0\le y\le 1}{\text{MAX}}\ f(x,y)\} =$$

$$\underset{0\le y\le 1}{\text{MAX}}\ \{\underset{0\le x\le 1}{\text{MAX}}\ f(x,y)\}.$$

We point out that the assumption of continuity can<u>not</u> de weakened to separate continuity as the following example shows.

<u>Example 1</u>

$$f(x,y) = \begin{cases} xy/(x^4 + y^4) & \text{if } (x,y)\ \varepsilon\ S - (0,0) \\ 0 & \text{if } (x,y) = (0,0) \end{cases}$$

<u>Theorem 2</u> Let $f(x,y)$ be continuous on the unit square S. For each $a\ \varepsilon\ [0,1]$, define $h(a) = \text{MAXIMUM}\ f(a,y)$. Then the function $h: [0,1] \to R$ is continuous.

<u>Proof</u>

Let $a\ \varepsilon\ [0,1]$ and let $\varepsilon > 0$ be given. Uniform continuity of $f(x,y)$ implies the existence of $\delta = \delta(\varepsilon) > 0$ such that $|f(a,y) - f(x,y)| < \varepsilon$ whenever $|x - a| < \delta$. Take x such that $|x - a| < \delta$ and let the maximum of $f(a,y)$ over y occur at $\bar{y}$ and let the maximum of $f(x,y)$ over y occur at $\bar{\bar{y}}$. That is, $h(a) = \underset{0\le y\le 1}{\text{MAXIMUM}}\ f(a,y) =$ $f(a,\bar{y})$ and $h(x) = \underset{0\le y\le 1}{\text{MAXIMUM}}\ f(x,y) = f(x,\bar{\bar{y}})$.

Then $|f(a,\bar{y}) - f(x,y)| < \varepsilon$ and $|f(a,\bar{\bar{y}}) - f(x,\bar{\bar{y}})| < \varepsilon$

$f(a,\bar{y}) < f(x,\bar{y}) + \varepsilon \le f(x,\bar{\bar{y}}) + \varepsilon$ and $f(a,\bar{\bar{y}}) - f(x,\bar{\bar{y}}) > -\varepsilon$

$f(a,\bar{y}) - f(x,\bar{\bar{y}}) < \varepsilon$ and $f(a,\bar{y}) - f(x,\bar{\bar{y}}) > -\varepsilon$.

Thus $|f(a,\bar{y}) - f(x,\bar{\bar{y}})| < \varepsilon$ whenever $|x - a| < \delta$ or $|h(a) - h(x)| < \varepsilon$ whenever $|x - a| < \delta$. This shows $h: [0,1] \to R$ is <u>uniformly</u> continuous on $[0,1]$.

The example cited earlier shows that the assumption of continuity on $f(x,y)$ in Theorem 2 can<u>not</u> be relaxed to separate continuity.

<u>Example 2</u>

$$f(x,y) = \begin{cases} xy/(x^4 + y^4) & \text{if } (x,y) \in S - \{(0,0)\} \\ 0 & \text{if } (x,y) = (0,0). \end{cases}$$

It is easily verified that here the function $h(a) = \underset{0 \leq y \leq 1}{\text{MAXIMUM}}\, f(a,y)$ is given by:

$$h(a) = \begin{cases} \dfrac{3}{4(\sqrt[4]{3}\ a^2)} & \text{if } 0 < a \leq 1 \\ 0 & \text{if } 0 = a. \end{cases}$$

That is, $h(a)$ is <u>not</u> continuous at $a = 0$.

The next theorem appeared as Problem E 2854 and its solution in the April 1982 issue of the <u>American Mathematical Monthly</u>.

<u>Theorem 3</u> Let $f(x,y)$ be a real-valued continuous function on the unit square $S = \{(x,y) : 0 \leq x,y \leq 1\}$. Additionally, suppose that for each $a \in [0,1]$, the maximum of $f(a,y)$ over y occurs at ONLY ONE value of y, say $\underset{0 \leq y \leq 1}{\text{MAXIMUM}}\, f(a,y) = f(a,y^*(a))$. Then the assignment $a \mapsto y^*(a)$ defines a continuous function $y^* : [0,1] \to [0,1]$.

<u>Proof</u> The proof is by contradiction. Suppose y^* is not continuous at some $a \in [0,1]$. Let $\{a_n\}$ be a sequence in $[0,1]$ convergent to a such that $b_n = y^*(a_n)$ fails to converge to $y\ (a)$. Since $\{b_n\}$ is a sequence in a compact space, we may assume, without loss of generality,

that $\{b_n\}$ converges to some number $b \in [0,1]$ (otherwise we select a convergent subsequence). Let $f(a,y^*(a)) - f(a,b) = \varepsilon$. Since $f(a,y^*(a)) = \underset{0 \le y \le 1}{\text{MAXIMUM}}\ f(a,y)$ we see that $f(a,y^*(a)) \geq f(a,b)$; i.e., $\varepsilon \geq 0$. The uniqueness of the maximum implies that $\varepsilon > 0$. For if $\varepsilon = 0$ then $f(a,y^*(a)) = f(a,b)$ or BOTH $y^*(a)$ AND b maximize $f(a,y)$ and $y^*(a) = b$, violating the assumed uniqueness of the maximum. We have:

$$|f(a,y^*(a))-f(a,b)| \leq |f(a,y^*(a))-f(a_n,b_n)| + |f(a_n,b_n)-f(a,b)|$$
$$= |h(a) - h(a_n)| + |f(a_n,b_n)-f(a,b)|$$

where h is as defined in Theorem 2.

Theorem 2 together with the fact that $a_n \to a$ implies that $|h(a) - h(a_n)| < 1/2\varepsilon$ whenever n is sufficiently large. Also, continuity of $f(x,y)$ together with the convergences $a_n \to a$ and $b_n \to b$ implies $|f(a_n,b_n) - f(a,b)| < 1/2\varepsilon$ whenever n is sufficiently large. Thus taking n so large that BOTH $1/2\varepsilon$-inequalities hold simultaneously we obtain the following contradiction:

$$f(a,y^*(a)) - f(a,b)| < 1/2\varepsilon + 1/2\varepsilon$$
$$\varepsilon < \varepsilon.$$

We acknowledge our gratitude to Dr. Charles Giel (formerly of A&T State University) for the proof of Theorem 3 above.

In a private communification, Professor R. A. Struble of North Carolina State University, gave the following solution to Problem E 2854 and hence an independent proof of Theorem 3.

<u>Alternate Proof of Theorem 3</u> (Direct Proof)

Let a [0,1] be given and let $\{a_n\}$ be a sequence in [0,1] such $a_n \to a$. We show $y^*(a_n) \to y^*(a)$. The sequence $\{y^*(a_n)\}$ is in the

compact space [0,1] and hence we may assume that $\{y^*(a_n)\}$ is convergent to some number $b \in [0,1]$ (otherwise we select a convergent subsequence). Continuity of $f(x,y)$ implies that $f(a_n,y^*(a_n)) \to f(a,b)$ and $f(a_n,y^*(a)) \to f(a,y^*(a))$. From the definition by y^*, it follows $f(a\ ,y^*(a_n)) \geq f(a_n,y^*(a))$. Thus
$\lim f(a_n,y^*(a_n)) \geq \lim f(a_n,y^*(a))$ or $f(a,b) \geq f(a,y^*(a))$. The last inequality says b maximizes $f(a,y)$ over y so that uniqueness of the maximum now implies $b = y^*(a)$; i.e., $y^*(a_n) \to y^*(a)$.

The proof of Theorem 3 published in the American Mathematical Monthly is shorter than either of the proofs given here; however the published proof relies on a compact graph theorem and, in our opinion is less instructive. Problem E 2854 asks if Theorem 3 may be generalized as follows. Suppose the requirement of the uniqueness of the maximum is no longer imposed and the function $y^* : [0,1] \to [0,1]$ is modified so that $y^*(a) = \text{MIN}\ \{y : y \text{ maximizes } f(a,y)\}$. Does the assignment $a \mapsto y^*(a)$ define a continuous function $y^* : [0,1] \to [0,1]$? The answer is NO! The following counterexample is given in the American Mathematical Monthly.

Example 3

$$f(x,y) = (x-1/2)(y-1/2). \quad \text{Here } y^*(a) = \begin{cases} 0 & : a \leq 1/2 \\ 1 & : a > 1/2 \end{cases}$$

Professor J. G. Mauldron of Amherst College points out that the function of Example 3 is unsatisfactory because it fails to satisfy the uniqueness property miserably at $a = 1/2$ in the sense that the set $\{y : y \text{ maximizes } f(1/2,y)\} = [0,1]$ and offers the following example instead.

Example 4

$$f(x,y) = (x-y)^2. \quad \text{Here } y^*(a) = \begin{cases} 1 & : a < 1/2 \\ 0 & : a \geq 1/2. \end{cases}$$

For the function $f(x,y)$ of Example 4, the departure from the uniqueness condition is MINIMAL in the sense that the $\{y : y \text{ maximizes } f(a,y)\}$ is a singleton for $a \neq 1/2$, while the set $\{y : y \text{ maximizes } f(1/2,y)\} = \{0,1\}$.

Professor Mauldron offers the following example to illustrate that the continuity requirement on $f(x,y)$ in Theorem 3 can<u>not</u> be relaxed to separate continuity.

<u>Example 5</u>

$$f(x,y) = \begin{cases} y & \text{if } x = 0 \\ 8y(x-y)/x^2 & \text{if } x \neq 0 \end{cases}$$

The function $f(x,y)$ satisfies the uniqueness condition but is only separately continuous. The induced function $y^*(a)$ is discontinuous at $a = 0$:

$$y^*(a) = \begin{cases} 1 & \text{if } a = 0 \\ 1/2a & \text{if } 0 < a \leq 1 \end{cases}$$

Looking at Examples 3 and 4, one may be tempted to conjecture that $y^* : [0,1] \to [0,1]$ enjoys the property of one-sided continuity. Professor Richard Tucker of A&T State University gives the following counter-example.

<u>Example 6</u>

$$F(x,y) = \begin{cases} f(x,y) : 0 \leq x \leq 1/2,\ 0 \leq y \leq 1 \\ f(1-x,y) : 1/2 < x \leq 1,\ 0 \leq y \leq 1 \end{cases}$$

where $f(x,y)$ is as in Example 4 or $f(x,y) = |x - y|$.

Here $$y^*(a) = \begin{cases} 1 : 0 \leq a < 1/2 \\ 0 : a = 1/2 \\ 1 : 1/2 < a \leq 1. \end{cases}$$

COMPUTATIONAL EXAMPLES

Aaron Chew wrote the BASIC programs for use on Texas Instruments 99/4A personal computer with Extended Basic module and Peripheral Expansion System. We express our deep appreciation to Aaron for his programming assistance.

The TWO-VARIABLE PROGRAM is based on the following procedure. Let $f(x,y)$ be defined on the closed rectangle $R = \{(x,y): a \leq x \leq b;$ $c \leq y \leq d$. First use an Initial Grid on the rectangle obtained by putting evenly spaced points on the sides of the rectangle lying on the co-ordinate axes:

$$a = x_o < x_1 < x_2 < \ldots < x_M = b; \; c = y_o < y_1 < y_2 < \ldots < y_M = d$$

where $x_i = a + i\frac{(b-a)}{M}$ and $y_j = c + j\frac{(d-c)}{M}$. The procedure first produces an initial approximation $(\bar{x},\bar{y})$ based on the points (x_i,y_j) of the initial grid. The Main Program then uses $(\bar{x},\bar{y})$ as STARTING POINT and procedes as follows. Fix $x = \bar{x}$ and minimize $f(\bar{x},y)$ over $y \; \varepsilon \; [c,d]$ using evenly spaced partition of the type used in the one-variable case; namely, evenly spaced points $(1/2)^N$ apart. Say $\underset{y}{\mathrm{MIN}} \; f(\bar{x},y)$ occurs at $y = \bar{y}_1$. Next minimize $f(x,\bar{y}_1)$ over $x \; \varepsilon \; [a,b]$, again using points that are $(1/2)^N$ apart. Say $\underset{x}{\mathrm{MIN}} \; f(x,\bar{y}_2)$ occurs at $x = \bar{x}_2$. Refer to $(\bar{x}_2,\bar{y}_2)$ as the second CROSSING. Repeat as often as desired.

II. RANDOM SEARCH

The domain of the function is closed and bounded and it will always be possible to select the initial starting points at the boundary. All the examples discussed here are of functions whose domains are of the shape of hypercubes, $a_i \leq x_i \leq b_i$. Therefore, starting points may be taken as a_i, $i = 1,2,3,\ldots$ The next point may be taken as $a_i + \varepsilon$, where $\varepsilon = (b_i - a\ )/N$, if one decides to use N points to obtain the first minimum. It is not really important which formula is used to generate points over the domain, as long as those domains are searched repeatedly without duplication. We evaluate at the first N points just generated and store the minimum and the coordinates of the minimizing point. We repeat the procedure M times. Therefore, in all M minimum values are saved together with the coordinates of the minimizing points. All the generated points have to be tested whether they belong to the domain before they can be used. The essential features of the algorithm are indicated in the flow-diagram (Figure 1).

The M stored points should cluster around the minima. An illustration of this concept is shown in Figure 2. The main task of this procedure is to locate all the cluster groups. We have achieved only partial success in reaching this objective because of a problem described below:

If some of minima lie very near to each other, this procedure cannot separate the clusters, because the radius of the hypersphere which embrace these cluster points should be very small and therefore many points still remain outside of any hypersphere. These points which are outside give false cluster groups and thereby increase the function evaluations later tremendously. Let us take the

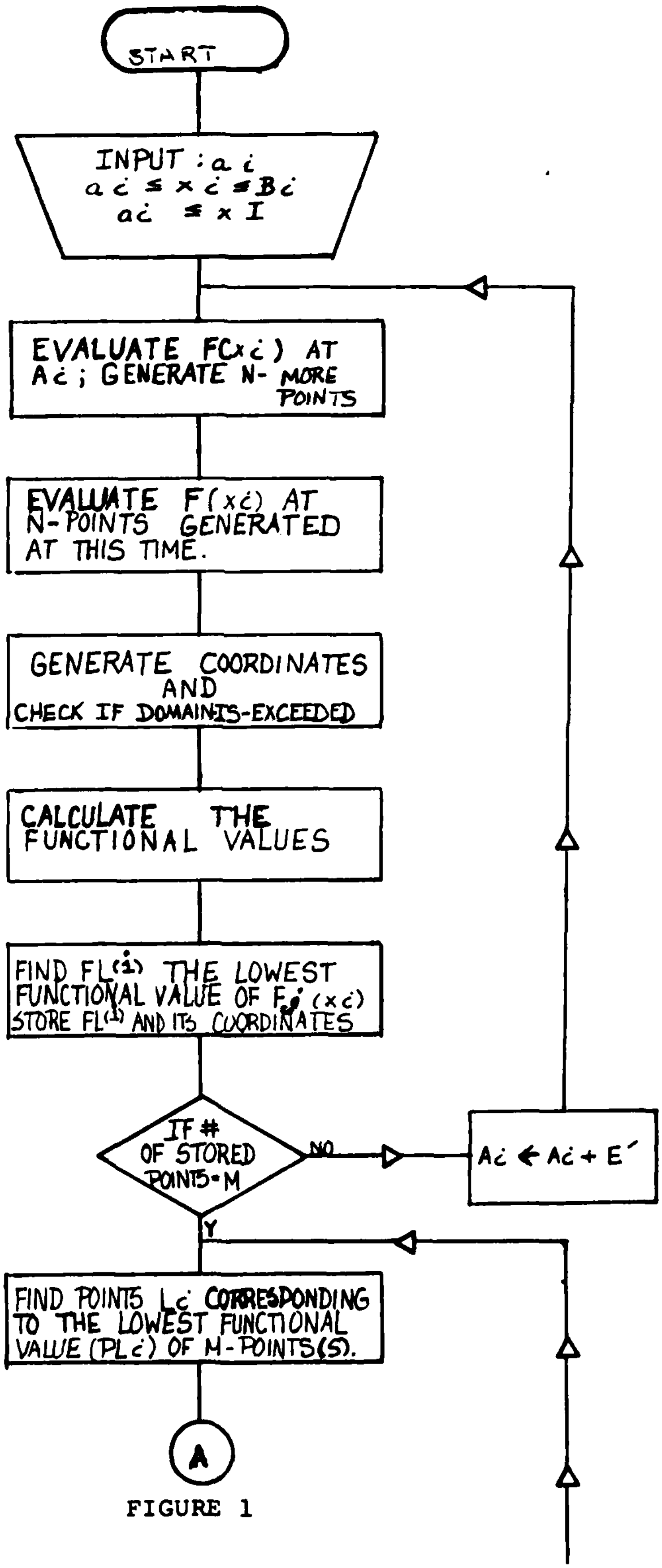

START
INPUT : a i
a i ≤ x i ≤ B i
a i ≤ x I
EVALUATE F(x i) AT
A i ; GENERATE N- MORE POINTS
EVALUATE F(x i) AT N-POINTS GENERATED AT THIS TIME.
GENERATE COORDINATES AND CHECK IF DOMAINS-EXCEEDED
CALCULATE THE FUNCTIONAL VALUES
FIND FL(i) THE LOWEST FUNCTIONAL VALUE OF Fj (x i) STORE FL(i) AND ITS COORDINATES
IF # OF STORED POINTS = M
NO
A i ← A i + E´
Y
FIND POINTS L i CORRESPONDING TO THE LOWEST FUNCTIONAL VALUE (PL i) OF M-POINTS(S).
A

FIGURE 1

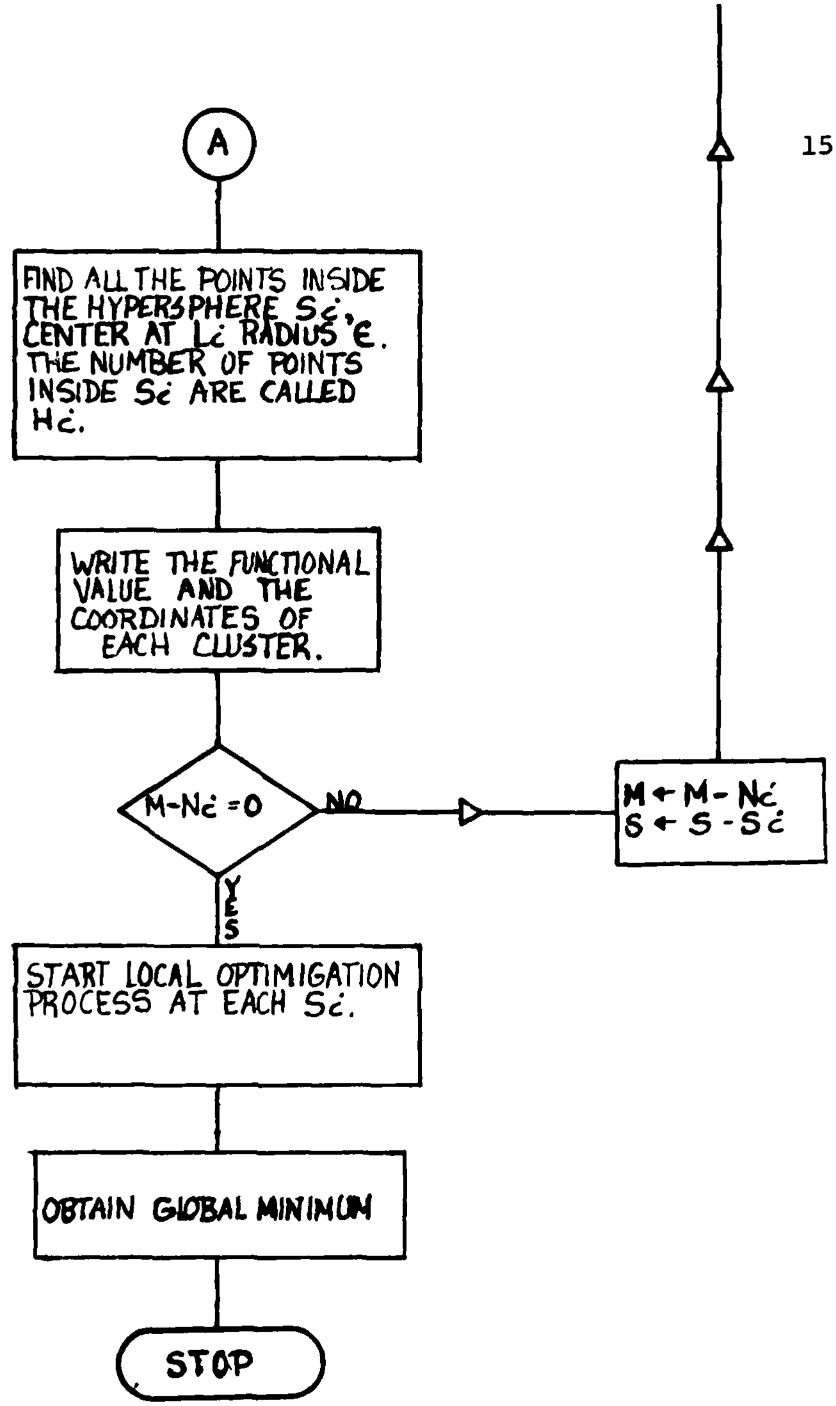

A
FIND ALL THE POINTS INSIDE THE HYPERSPHERE Sċ, CENTER AT Lċ RADIUS ϵ. THE NUMBER OF POINTS INSIDE Sċ ARE CALLED Hċ.
WRITE THE FUNCTIONAL VALUE AND THE COORDINATES OF EACH CLUSTER.
M-Nċ=0
NO
M ← M - Nċ
S ← S - Sċ
YES
START LOCAL OPTIMIGATION PROCESS AT EACH Sċ.
OBTAIN GLOBAL MINIMUM
STOP

FIGURE 1

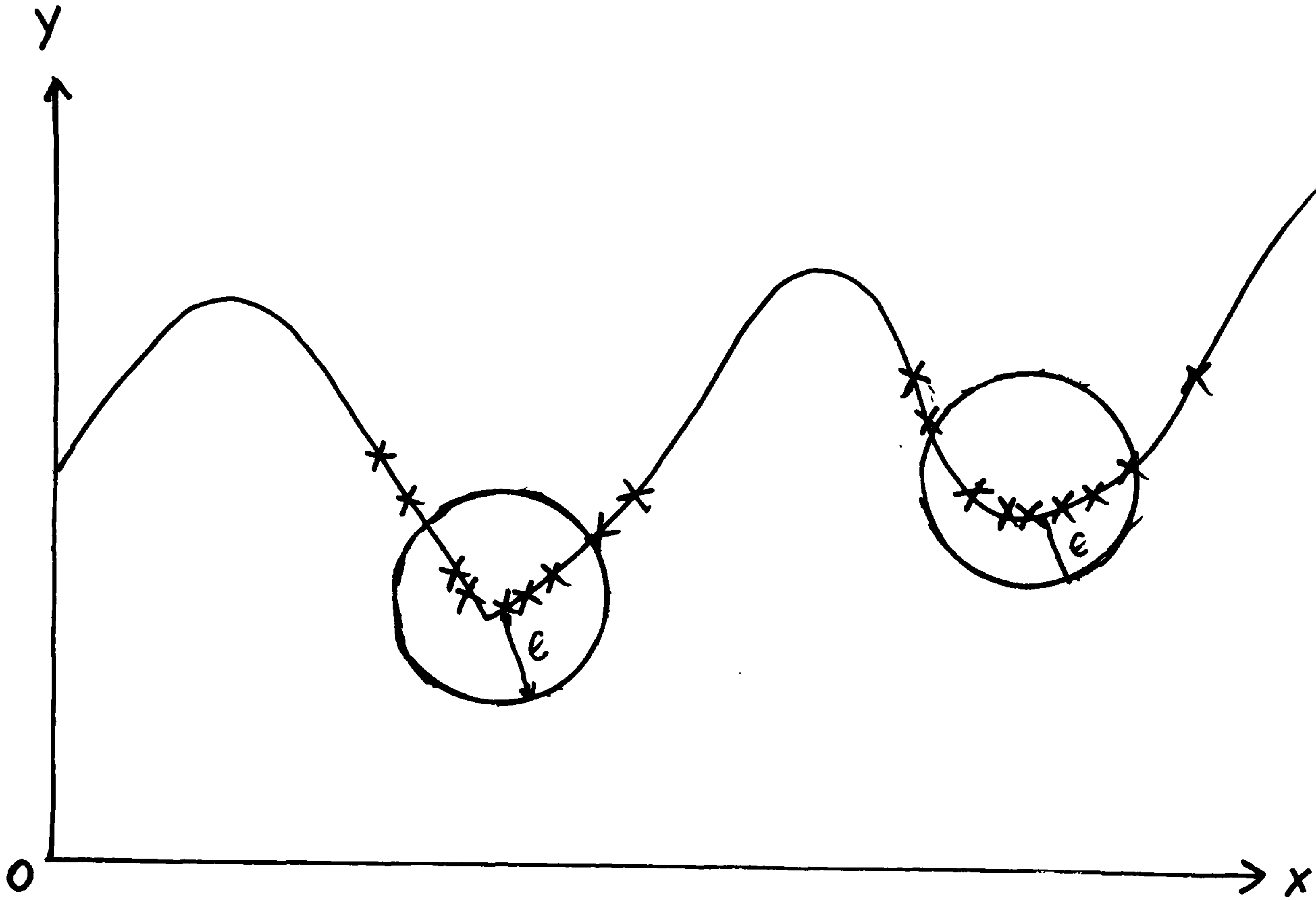


FIGURE 2

following function as example: $f(x_1,x_2) = (|x_1|-1)^2 + (|x_2|-2)^2$

$$-4 \le x_1, x_2 \le 4.$$

There are four minima with function value $f = 0$ and coordinates $(-1,-2),(-1,2),(1,-2),(1,2)$. All these clusters lie quite far apart and our procedure can obtain all of them very quickly. However, if $f(x_1,x_2) = (|x_1| - 0.1)^2 + (|x_2| - 0.5)^2$ this procedure will lead to one minimum point only since all four points (0.1,0.5), (0.1,-0.5),(-0.1,0.5) and (-0.1,-0.5) are lying on a very small rectangle.

After separating the cluster the next biggest task is to find the actual minimum in each cluster group. Any local optimizing method may be used. However, Nelder and Mead Simplex Search method is the most efficient one for non-differentiable functions. We have used Nelder and Mead Simplex Search method [4] in our program. The Nelder and Mead Simplex Search requires $m + 1$ points for m-dimensional space and they may not lie on the same hyperplane. Therefore, each cluster group, or the hyperspheres which embrace the cluster groups must include at least $m + 1$ points to start the initial simplex. So, not only is the counting of points necessary in each cluster but also sometimes the points must be regenerated if the points fall short.

The checking of collinearity is another important task in Simplex Search method. If the simplex repeats itself for a specific number of times, this has to be modified to prevent from collapsing the simplex. One way to solve this problem is to replace a point from the collapsing simplex by a point which lies on the orthogonal direction to the hyperplane.

The Choice of the Number of Retained Points:

The number of retained points may depend on the size of the domain and as well as the number of variables. Suppose we start with fifty points; fifty functional evaluations are performed and one point with lowest functional value is retained. If one wants to retain 50 points, 2500 function evaluations are required. Therefore, the number of function evaluations is very high where as storage requirement is comparatively less.

Constraints:

All the global optimization problems may be regarded as constrained in the sense that the search is confined within the initially prescribed domain. If any point falls out of this domain, that point has to be discarded. When additional constraints are imposed, then, depending on the number and complexity of these constraints, a sufficiently large number of points has to be selected to insure that a reasonable proportion of points from the totality of trial points be included.

The program is written in FORTRAN IV and several examples are discussed. Since we are using Simplex Search Method, the number of dimensions must be more than one. The program is attached in Appendix B.

Example 1

The function to be minimized is

$$f(x,y,z) = (x - y + z)^2 + (-x + y + z)^2 + (x + y - z)^2,$$

$$-1 \le x,\ y,\ z \le 1$$

It is easy to show that f is a strictly convex quadratic function with an unique minimum at (0,0,0) and f = 0.

After 2732 iterations, we have

$f = 0$

$x = 0$

$y = 0$

$z = 0$

Actual values: $f = 0$, $x = 0$, $y = 0$, $z = 0$

The method of systematic search takes 12096 iterations to arrive at this result.

In this connection it must be pointed out that in using the systematic search method, we have tried to adhere to <u>standardized</u> values for the number of initial grid points and the number of crossings. Since the function is NON-NEGATIVE and the actual optimal point is (0,0,0), the method of bisection would yield the answer on the very first bisection (27 evaluations at most!). Hence the computer operator would STOP the computer after ONLY 27 evaluations because he sees that f already attains 0 [and can never be improved] after 27 evaluations.

<u>Example 2</u>

This example is used to compare the result obtained by the method systematic search (discussed in this paper, Example 19, Table 3) and the actual values. The function is

$$f(x,y,z) = |x-1| + |y-1.5| + |6z-1|.$$

$$0 \le x,\ y \le 3,\ 0 \le z \le 1.5.$$

<u>Actual solution:</u>

$\mathrm{Min}(f(x,y,z) = 0$

$x = 1$, $y = 1.5$, $z = 0.1666...$

By the Systematic Search Method:

$$\text{Min}(f(x,y,z) = 0.00390625$$

$$x = 1,\ y = 1.5,\ z = 0.16605625$$

Number of evaluations: 18752

By the Random Search Method:

$$\text{Min}(f(x,y,z) = 0.000001326$$

$$x = 1,\ y = 1.5,\ z = 0.166667$$

Number of evaluations: 3008

Example 3

As another example, let us take the following function which was chosen by both Becker and Lago and Price's CRS algorithm (with additional constraint):

$$f(x_1,x_2,x_3) = 9-8x_1-6x_2-4x_3+2x_1^2+2x_2^2 +x_3^2+2x_1x_2+2x_1x_3$$

$$0 \le x_1x_2 \le 3,\ 0 \le x_3 \le 1.5.$$

The actual solution is

$f = 0$, $x_1 = 1$, $x_2 = 1$, $x_3 = 1$. The Random Search method achieves this solution in 2686 evaluations where $f = -0.1192\times10^{-06}$

$$x_1 = 1,\ x_2 = 0.9999,\ x_3 = 1$$

The method of systematic search takes 14144 evaluations.

Example 4

As a final example, we like to consider the following function to obtain all the four minima. Becker and Lago and Price also discussed a similar function. Their function was

$$f(x_1,x_2) = (|x_1| - 5)^2 + (|x_2| - 5)^2.$$

Price obtained all the four minima around $O(10^{-6})$ after 5000

evaluations but not obtained the coordinates. We take

$$f(x_1,x_2,x_3) = (|x_1| - 5)^2 + (|x_2| - 5)^2 + (x_3 - 1)^2$$

$$-10 \leq x_1,x_2,x_3 \leq 10.$$

All the four minima are obtained after 4010 evaluations:

Function Value	Coordinates		
	x_1	x_2	x_3
0.8298×10^{-10}	5.0	5.0	1.0
0.244×10^{-9}	-5.0	-5.0	1.0
0.1591×10^{-9}	-5.0	5.0	1.0
0.1699×10^{-9}	5.0	-5.0	1.0

Actual minimum is of course 0 at all these four points.

The computer printout of the unified program is enclosed in the Appendix B.

Conclusion:

The Random Search Method described in this paper is not really a Random Search. Besides the initial point - generation technique, everything later becomes more systematic than random. The method seems to be very efficient for problems wherever the Nelder and Mead Simplex Search method applies. It suffers a serious setback if some of the minima are very 'close' to each other. How close is very 'close'? This is an open question. One may use different local optimization techniques to avoid this situation. The problem of collapsing simplex may be handled as suggested in this paper.

Examining the program, one discovers that the storage requirement is not as great as it first appears. All the initial points

generated need not be saved. We need only to save the number of retained points which actually form clusters.

The work on the Method of Systematic Search has generated some mathematical theory and, it appears that more theoretical developments may be possible. Some of the advantages of the systematic search method are:

(1) it is applicable to functions of a single variable

(2) it is direct

(3) it is easy to execute (the problems of simplicial collapse, etc. do not appear)

(4) it is independent of other search procedures already in existence

(5) it goes after the global optimum without first calculating local optima

(6) it is not sensitive to 'nearness' of the local optima to each other

The method suffers from the standpoint of being computationally uneconomical in that the number of evaluations increases geometrically with an increase in the number of variables. Also the method of systematic search does not, in general, obtain <u>all</u> the local minima. This, in turn, may lead to some doubt as to where the actual global minimum occurs.

This is a serious problem attributed to all procedures which find global minimum without calculating derivatives such as the method of Becker and Lago and Price's Controlled Random Search Procedure. However, the method of Random Search appears to overcome this problem.

<u>Summary of Most Important Results</u>:

(a) The following three theorems have been established:

(1) <u>Theorem 1</u>: Let $f(x,y)$ be a real valued function which is

continuous on a compact domain D. Then

$$\max_{(x,y)\in D} f(x,y) = \max_{x\in D_y}\left\{\max_{y\in D_x} f(x,y)\right\}$$ where for each fixed x,

$D_x = \{y:(x,y)\in D\}$ and for each fixed y, $D_y = \{x:(x,y)\in D\}$.

(2) Theorem 2: Let f(x,y) be continuous on the unit square S. For each a ε [0,1], define $h(a) = \max_{0\le y\le 1} f(a,y)$. Then the function h: $[0,1] \to R$ is continuous.

(3) Theorem 3*: Let f(x,y) be a real valued continuous function on the unit square $S = \{(x,y): 0 \le x,y \le 1\}$. Additionally suppose that for each a ε [0,1], the maximum of f(a,y) over y occurs at only one value of y, say Max f(a,y) = f(a,y*(a)). Then the assignment $a \longmapsto y^*(a)$ defines a continuous function $y^*: [0,1] \to [0,1]$.

(b) A complete program to find the various cluster groups of a multimodal non-differentiable continuous function defined on a compact domain and to pinpoint the minimum value of the function at each cluster group using local optimizing technique is written.

List of All Participating Personnel:

1. Dr. Bolindra N. Borah, Professor, Department of Mathematics and Computer Science, N. C. A&T State University (Co-Principal Investigator).

2. Dr. James F. Chew, Associate Professor, Department of Mathematics and Computer Science, N. C. A&T State University (Co-Principal Investigator).

3. Mr. Duane D. Holmes, Student Assistant, Senior, Mathematics, N. C. A&T State University.

---

*This theorem appeared as problem E2854 and its solution in the April 1982 issue of the American Mathematical monthly.

4. Mr. Chester Terry, Student Assistant, Senior, Mechanical Engineering, N. C. A&T State University.

5. Mr. Pomorantz D. Sutton, Student Assistant, Junior, Computer Science, N. C. A&T State University.

5. Bibliography:

1. Brooks, S. H., A Discussion of Random Methods for Seeking Maxima Operation Research, Vol. 6, pp. 244-251, (1958).

2. Becker, R. W., and Lago, G. V., A Global Optimization Algorithm, Proceeding of the Eighth Allerton Conference on Circuits and System Theory, (1970).

3. Price, W. L., A Controlled Random Search Procedure for Global Optimization, Computer Journal, Vol. 20, No. 4 (1977).

4. Nelder, J. A., and Mead, R., A Simplex Method for Function Minimization, The Computer Journal, Vol. 7, pp. 308-313 (1965).

NOTE: We could not obtain the paper from J. J. Bryan, S. J. Dwyer and G. V. Lago (1969), so no reference is mentioned in this work.

#	function (f)	DOMAIN	COMPUTED SOLUTION	ACTUAL SOLUTION
1	MIN $(100(x-x^2)^2 + (6.4(x-.5)^2 - x - .6)^2)$	$0 \le x \le .5$	$f = .6428737641$ $x = .05078125$ $N = 10$; 512 evaluations	x between .0507 & .0508
2	MAX $(1 - x^2 + \sin x)$	$0 \le x \le .5$	$f = 1.232465575$ $x = .4501953125$ $N = 10$; 512 evaluations	x between .450 & .451
3	MAX $(-3x^3 + 3x^2 + x)$	$0 \le x \le 2$	$f = 1.1840949$ $x = .8046875$ $N = 10$; 2048 evaluations	$f = 1.1840949$ $x = \frac{1+\sqrt{2}}{3} \approx .8047378541$
4	MAX $(x \cos x)$	$0 \le x \le 1.6$	$f = .561096$ $x = .8609375$ $N = 10$; 1639 evaluations	
5	MAX $(e^{-x} \cos (x + 1/4\pi))$	$4 \le x \le 9$	$f = .0063522$ $x = 4.712890625$ $N = 10$; 5120 evaluations	$f = .0063522$ $x = \frac{3\pi}{2} \approx 4.712389$
6	MAX $(2 - \|3x - 1\|)$	$0 \le x \le 1$	$f = 1.9902343$ $x = .3330078125$ $N = 10$; 1024 evaluations	$f = 2$ $x = 1/3 = .333 \ldots$
7	MAX $\frac{2x^2}{(x+1)(x-2)}$	$-1/2 \le x \le 1$	$f = 0$ $x = 0$ $N = 10$; 1536 evaluations	$f = 0$ $x = 0$

TABLE 1

#	function (f)	DOMAIN	COMPUTED SOLUTION	ACTUAL SOLUTION
8	MAX $(-2x^2)/(x+1)(x-2)$	$-10 \le x \le -2$	f = -1.777777778 x = -4 N = 10; 8192 evaluations	$f = -16/9 = -1.777\ldots$ x = -4
9	MAX $(x^2 + 3x + 2)/(x+3)(x-1)$	$-2 \le x \le -1$	f = .0669873 x = -1.5361322813 N = 10; 1024 evaluations	f = .669873 $x = -5 + 2\sqrt{3}$ = -1.5358984
10	MIN $100(y-x^2)^2 + (6.4(y-.5)^2 - x - .6)^2$	$0 \le x,y \le 1$ $y \le x$ $x + y \le 1$	f = .0000693987 x = .34 y = .115234375; M = 50; N = 9 CROSS = 3; 4036 evaluations	f = 0 x = .3414 y = .116554
11	MIN $100(y-x^2)^2 + (6.4(y-.5)^2 - x - .6)^2$	$0 \le x,y \le 1$ $x \le y$ $x + y \le 1$	f = .642941 x = .05 = y; M = 60; N = 9 CROSS = 3; 5136 evaluations	f(.05075) = .6429119 x = y between .0507 & .0508
12	MIN $1 + \sin^2 x + \sin^2 y - .1\, e^{-(x^2+y^2)}$	$-1/2 \le x,y \le 1/2$	f = .9 x = 0 = y; M = 50; N = 9 CROSS = 3; 4036 evaluations	f = .9 x = 0 = y
13	MAX $(x+y)/(x^2 + y^2 + 1)$	$0 \le x,y \le 1$ $0 \le x,y \le 1$	f = .707106 x = .70703125 = y; M = 50; N = 9 CROSS = 3; 4036 evaluations	$f = \frac{1}{\sqrt{2}} \approx .7071068$ $x = \frac{1}{\sqrt{2}} \approx .7071068 \approx y$
14	MAX $\sin(\pi x) + \sin(\pi y) + \sin(\pi(x+y))$		f = 2.598065347 x = .333984372 y = .33203125; M = 50; N = 9 CROSS = 3; 4036 evaluations	f = 2.5980762 $x = 1/3 = .333\ldots$ $y = 1/3 = .333\ldots$

TABLE 2

#	function (f)	DOMAIN	COMPUTED SOLUTION	ACTUAL SOLUTION
15	MAX $(9y - 32x - y^3 - x^4)$	$-3 \le x \le 0$ $0 \le y \le 2$	f = 58.39230413 x = -2 y = 1.732; M = 50; N = 9 CROSS = 3; 9668 evaluations	f = 58.392305 x = -2 $y = \sqrt{3} \approx 1.7320508$
16	MAX $(\lvert x-y\rvert + (\lvert x-1 - y)^2)$	$-2 \le x \le 2$ $-1 \le y \le 3$	f = 17 x = -2 y = -1; M = 60; N = 9 CROSS = 3; 14788 evaluations	
17	MAX $(xy)/(x^2 + y^2)$	$1 \le x \le 2$ $0 \le y \le 1$	f = .5 x = 1 = y; M = 60; N = 9 CROSS = 3; 5136 evaluations	
18	MIN $(9-8x-6y-4z+2x^2+2y^2+z^2+2xy+2xz)$	$0 \le x,y,z \le 1.5$ $x+y+2z \le 3$	f = .1125 x = 1.35; M = 20; N = 9 y = .75; CROSS = 3 z = .45; 14144 evaluations	f = 1/9 = .111... x = 4/3 = 1.333... y = 7/9 = .777... z = 4/9 = .444...
19	MIN $(\lvert x-1 + \lvert y-1.5\rvert + \lvert 6z-1\rvert)$	$0 \le x,y \le 3$ $0 \le z \le 1.5$ $x+y+2z \le 3$	f = .00390625 x = 1 ; M = 20; N = 9 y = 1.5; CROSS = 3 z = .16605625; 18752 evaluations	f = 0 x = 1 y = 1.5 z = 1/6 = .1666...
20	MIN $(9-8x-6y-4z+2x^2+2y^2+z^2+ 2xy+2xz)$	$0 \le x,y,z \le 1.5$	f = .0000953674 x = 1.013671875; M = 20; N = 9 y = .9921875 ; CROSS = 3 z = .986328125 ; 14144 evaluations	f = 0 x = 1 y = 1 z = 1
21	MIN $((x-y+z)^2+ (-x+y+z)^2 + (x+y-z)^2)$	$-1 \le x,y,z \le 1$	f = 0 ; M = 20; N = 9 x = y = z = 0 ; CROSS = 1 12096 evaluations	f = 0 x = y = z = 0

TABLE 3

6. Appendixes:

APPENDIX A

PROGRAM ONE (VARIABLE) (See Example #1)

```
1    CONS = 1.E 20
10   INPUT "N" : N
20   INPUT "START & END POINT" : B,E
30   FOR X = B TO E STEP .5 ∧ N
40   A = 100*(X -X ∧ 2) ∧ 2 + [6.4*(X-.5) ∧ 2 - X - .6] ∧ 2 :: GOSUB 100
50   NEXT X
55   GOTO 110
100  IF A < CONS THEN CONS = A :: X0 = X :: PRINT "ANSWER"; CONS ::
     PRINT "X"; X0 :: PRINT :: RETURN ELSE RETURN
110  END
```

PROGRAM TWO (VARIABLE) (See Example #10)

X = ROW ; Y = COLUMN

```
5         INPUT "INIT GRID?" : IG :: CONS = 1. E + 100
15        INPUT "CROSSING" : LOOP :: INPUT "N" : CUTTER :: INPUT
          "ROW BEGINNING" : RB :: INPUT "ROW END" : RE :: INPUT
          "COLUMN BEGINNING" : CB :: INPUT "COLUMN END" : CE
20        R1 = RB :: GOSUB 80
30        FOR L123 = 1 TO LOOP
35        FOR COL = CB TO CE STEP .5 ∧ CUTTER :: ANSWER = 100*
          ((COL-R1 ∧ 2) 2) + (6.4* ((COL - .5) ∧ 2) - R1 - .6) ∧ 2 ::
          GOSUB 65 :: NEXT COL
45        FOR ROW = RB TO RE STEP .5 ∧ CUTTER :: ANSWER = 100*
          ((C1 - ROW ∧ 2) ∧ 2) + (6.4*((C1 - .5) ∧ 2 - ROW - .6) ∧ 2 ::
          GOSUB 70 :: NEXT ROW
55        NEXT L123
60        GOTO 100
65        IF ANSWER < CONS AND R1 ≥ COL AND COL + R1 ≤ 1 THEN
          CONS = ANS :: C1 = COL :: PRINT "ANSWER" ; CONS ::
          PRINT "ROW" ; R1 :: PRINT "COLUMN" ; C1
66        RETURN
70        IF ANSWER < CONS AND ROW ≥ C1 AND C1 + ROW ≤ 1 THEN CONS =
          ANSWER :: R1 = ROW :: PRINT "ANSWER" : CONS :: PRINT
          "ROW" ; R1 :: PRINT "COLUMN" ; C1
71        RETURN
```

PROGRAM TWO (VARIABLE) (Continued)

```
75        IF ANSWER < CONS AND ROW ≥ COL AND ROW + COL ≤ 1 THEN
          CONS = ANSWER ::
R1 = ROW  : : C1 = COL :: PRINT "ANSWER" ; CONS :: PRINT "ROW" ;
          R1 :: PRINT "COLUMN" : C1
76        RETURN
80        FOR COL = CB TO STEP (CE-CB)/IG
85        FOR ROW = RB TO RE STEP (RE-RB)/IG :: ANSWER = 100*
          ((COL-ROW ∧ 2) ∧ 2) + (6.4* ((COL - .5) ∧ 2 - ROW - .6) ∧ 2
          :: GOSUB 75 :: NEXT ROW
90        NEXT COL
95        RETURN
100       END
```

PROGRAM THREE (VARIABLE) (See Example #19)

X = DEPTH; Y = COLUMN ; Z = ROW

```
1     CONS = 1.E 10
10    CALL CLEAR
20    INPUT "INITIAL GRID" : IG
30    INPUT "ROW START & END" : RB,RE
40    INPUT "COLUMN START & END" ; CB,CE
50    INPUT "DEPTH START & END" : DB,DE :: INPUT "LOOP":L
60    INPUT "N" : N
70    FOR DEPTH = DB TO DE STEP (DE-DB)/IG
80    FOR COL = CB TO CE STEP (CE-CB)/IG
90    FOR ROW = RB TO RE STEP (RE-RB)/IG :: A = ABS(DEPTH-1) +
      ABS(COL - 1.5) + ABS(6* ROW - 1) :: GOSUB 500 :: NEXT ROW
100   NEXT COL :: NEXT DEPTH
105   PRINT "OUT OF SUBPROGRAM"
106   FOR L1 = 1 TO L
110   FOR DEPTH = DB TO DE STEP .5^N :: A = ABS(DEPTH - 1) +
      ABS(COLO - 1.5) + ABS(6* ROWO - 1) :: GOSUB 600 :: NEXT DEPTH
120   FOR COL = CB TO CE STEP .5^N :: A = ABS(DEPTHO - 1) +
      ABS(COL - 1.5) + ABS(6* ROWO - 1) :: GOSUB 700 :: NEXT COL
130   FOR ROW = RB TO RE STEP .5^N :: A = ABS(DEPTHO - 1) +
      ABS(COLO - 1.5) + ABS(6* ROW - 1) :: GOSUB 800 :: NEXT ROW
134   PRINT L1
135   NEXT L1
140   END
```

PROGRAM THREE (VARIABLE) (Continued)

```
500   IF < CONS AND DEPTH + COL + 2* ROW ≤ 3 THEN DEPTHO = DEPTH
      :: ROWO = ROW :: COLO = COL :: CONS = A :: GOSUB 1000 ::
      RETURN ELSE RETURN
600   IF A < CONS AND DEPTH + COLO + 2* ROWO ≤ 3 THEN CONS = A ::
      DEPTHO = DEPTH :: GOSUB 1000 :: RETURN ELSE RETURN
700   IF A < CONS AND DEPTHO + COL + 2* ROWO ≤ 3 THEN COLO = COL
      :: CONS = A GOSUB 1000 :: RETURN ELSE RETURN
800   IF A < CONS AND DEPTHO + COLO + 2* ROW ≤ 3 THEN CONS = A ::
      ROWO = ROW :: GOSUB 1000 :: RETURN ELSE RETURN
1000  PRINT "ANSWER"; CONS: "DEPTH";DEPTH:"COLUMN";COLO:"ROW";ROWO
      :: PRINT :: RETURN
```

APPENDIX B

COMPUTER PRINT OUT:RANDOM SEARCH

```
 P
C        FIND ABSOLUTE MINIMUM OF NON-DIFFERENTABLE BOUNED
C        FUNCTION WITH OR WITHOUT CONSTRAINT
         DIMENSION A(3,50,50),FMAX(50),CMAX(50,3),F(50,50),CF(50,
3),
      1  MAX(50,50),X(50),Y(50),YMAX(50,50),CYMAX(50,50,3),FMAX1(
50),
      2  P(3),CMAX1(50,3),CFAX(50,3),CL1(50,50),FCL(50),CFCL(50,3
),
      3  CMAX2(50,3),FMAX2(50),CL(50,50,3),D(6),
      4  E(3)
         N=20
         I1=6
         I2=3
         OPEN(UNIT=20,FILE='IN1.DAT')
         OPEN(UNIT=21,FILE='RES.OUT')
         IUI=20
         IU=5
C        READ DOMAIN PARAMETER
         READ(IUI,*)(D(I),I=1,6)
*P

   5     FORMAT(6F5.1)
         RAD=4
         KOUNT=0

C        INITALIZE THE ARRAY
         N=50
         DO 7 K=1,I2,1
   7     A(K,1,1)=D(K)
C        CALCULATE EPSILON VALUES
C
         DO 8 I=1,I2,1
   8     E(I)=(D(I+3)-D(I))/50.0
C
C        GENERATE COORDINATES
C
*P
         DO 40 J=1,N,1
         DO 30 I=1,N,1
         DO 9 K=1,I2,1
9        A(K,I,J)=A(K,1,1)*(-1)**(I*J*K)+I*E(K)*(-1)**I+J*E(K)*(-
1)
      1  **J+K*E(K)*(-1)**K
C
C        CHECK IF EXCEEDS DOMAIN
C
         DO 25 K=1,I2,1
         IF(A(K,I,J).LE.D(K)) A(K,I,J)=D(K+3)-J*E(K)
         IF(A(K,I,J).GE.D(K+3)) A(K,I,J)=D(K)+J*E(K)
   25    CONTINUE
C
C        CALCULATE  THE FUNCTIONAL VALUES
C
         N1=J
*
```

```
 P
                K1=I
                F(I,J)=CAL(A,K1,N1)
                KOUNT=KOUNT+1
      C         WRITE(IU,26)(A(K,I,J),K=1,I2),F(I,J)
        26      FORMAT(4X,3(E10.4,4X),4X,E10.4/)
        30      CONTINUE
      C
      C         CALL SUBROUTINES TO FIND MIN OF THE 50 POINTS JUST EVALA
TED.
      C
                K=J
                I3=I2
                CALL FMAXI(A,F,FMAX,CMAX,K,N,I3)
      C
      C         CALCULATION TO START NEXT SET OF POINTS
      C
                AJ=J
*P
                AJ=AJ/50.
                DO 35 K=1,I2,1
        35      A(K,1,1)=A(K,1,J)+AJ*(-1)**J
         40     CONTINUE
      C
      C         OUTPUT MINIMUN VALUE & COORDINATES
      C
                WRITE(IU,45)
        45      FORMAT(//,5X,'VALUE OF THE FUNCTION',5X,'FIRST COORDINAT
E',5X,'SECOND
           1COORDINATE',5X,'THIRD COORDINATE',/)
                DO 50 I=1,N,1
       50       WRITE(IU,55) FMAX(I),(CMAX(I,K),K=1,I2)
        55      FORMAT(5X,E11.5,5X,E11.5,5X,E11.5,5X,E11.5)
      C
      C         CALL PLOTTING ROUTINE
      C
*P
                DO 200 J=1,4,1
                JJ=J
                CALL MAX2(CMAX,FMAX,FMAX2,CMAX2,N,JJ,I2)
                GO TO (60,62,64,67)J
       60       WRITE(IU,61)J
       61       FORMAT(10X,' GROUP',I3)
                WRITE(IU,65) FMAX2(J),(CMAX2(J,K),K=1,I2)
        65      FORMAT(5X,E12.4,E12.4,E12.4,E12.4,/)
                GO TO 80
       62       WRITE(IU,61)J
                WRITE(IU,65)FMAX2(J),(CMAX2(J,K),K=1,I2)
                GO TO 80
       64       WRITE(IU,61) J
                WRITE(IU,65) FMAX2(J),(CMAX2(J,K),K=1,I2)
                GO TO 80
       67       WRITE(IU,61)J
*
```

```
 P
             WRITE(IU,65)FMAX2(J),(CMAX2(J,K),K=1,I2)
      80     K=0
             DO 84 I=1,N,1
             IF(FMAX(I).EQ.9.1E+10)GO TO 84
             DIST=0.0
             DO 81 KI=1,I2,1
             P(KI)=CMAX2(J,KI)-CMAX(I,KI)
      81     DIST=DIST+(P(KI)**2)
             Q=SQRT(DIST)
             IF(Q.GE.RAD) GO TO 84
             K=K+1
             FCL(K)=FMAX(I)
             DO 82 KI=1,I2,1
      82     CFCL(K,KI)=CMAX(I,KI)
             FMAX(I)=9.1E+10
             DO 83 KI=1,I2,1
*P
      83     CMAX(I,KI)=9.1E+10
      84     CONTINUE
             K=K+1
             FCL(K)=FMAX2(J)
             DO 85 KI=1,I2,1
      85     CFCL(K,KI)=CMAX2(J,KI)
             FMAX2(J)=9.1E+10
             IF(K .EQ. 0.) GO TO 220
             GO TO(101,111,121,131)J
     101     K1=K
             DO 106 I=1,K1,1
             CL1(J,I)=FCL(I)
             FCL(I)=0.
             DO 105 KI=1,I2,1
             CL(J,I,KI)=CFCL(I,KI)
     105     CFCL(I,KI)=0.
*P
     106     CONTINUE
             GO TO 200
     111     K2=K
             DO 116 I=1,K2,1
             CL1(J,I)=FCL(I)
             FCL(I)=0.
             DO 115 KI=1,I2,1

             CL(J,I,KI)=CFCL(I,KI)
     115     CFCL(I,KI)=0.
     116     CONTINUE
             GO TO 200
     121     K3=K
             DO 126 I=1,K3,1
             CL1(J,I)=FCL(I)
             FCL(I)=0.
*
```

```
P
             DO 125 KI=1,I2,1
             CL(J,I,KI)=CFCL(I,KI)
      125    CFCL(I,KI)=0.
      126    CONTINUE
             GO TO 200
      131    K4=K
             DO 136 I=1,K4,1
             CL1(J,I)=FCL(I)
             FCL(I)=0.
             DO 135 KI=1,I2,1
             CL(J,I,KI)=CFCL(I,KI)
      135    CFCL(I,KI)=0.
      136    CONTINUE
      200    CONTINUE
    C
    C        WRITE THE FUNCTION-VALUE AND THE COORDINATES OF THE WHOL
E CLUSTER
*P
    C
      220    IF(K1.EQ.0) GOTO 225
             WRITE(IU,222) K1
      222    FORMAT(10X,' CLUSTER ',I3)
             DO 224 I=1,K1,1
      224    WRITE(IU,*) CL1(1,I),(CL(1,I,KI),KI=1,I2)
      225    IF(K2.EQ.0) GOTO 230
             WRITE(IU,226)K2
      226    FORMAT(10X,'CLUSTER ',I3)
             DO 228 I=1,K2,1
      228    WRITE(IU,*) CL1(2,I),(CL(2,I,KI),KI=1,I2)
      230    IF(K3.EQ.0) GO TO 235
             WRITE(IU,232)K3
      232    FORMAT(10X,'CLUSTER ',I3)
             DO 234 I=1,K3,1
      234    WRITE(IU,*) CL1(3,I),(CL(3,I,KI),KI=1,I2)
*P
      235    IF(K4.EQ.0) GO TO 240
             WRITE(IU,236) K4
      236    FORMAT(19X,'CLUSTER ',I3)
             DO 238 I=1,K4,1
      238    WRITE(IU,*) CL1(4,I),(CL(4,I,KI),KI=1,I2)
      240    WRITE(IU,250)
      250    FORMAT(5X,'MAX FUNCT  ',5X,'COORD TE-1',5X,'COORD TE-2',
5X,
            1 'COORD TE-3',//)
             KK=0
             DO 350 J=1,4,1
             GO TO (302,306,310,314)J
      302    IF(K1.EQ.0) GO TO 350
             J1=K1
             I=J
             ICOUNT=0
      303    CALL SMPLEX(CL1,CL,CYMAX,YMAX,D,I,J1,I2,ICOUNT)
*
```

```
 P
                WRITE(IU,305)ICOUNT
      305       FORMAT(2X,'GROUP ITERATION',2X,I4)
                KK=KK+ICOUNT
                WRITE(IU,304) YMAX(J,1),(CYMAX(J,1,KI),KI=1,I2)
      304       FORMAT(5X,E10.4,3(6X,E10.4),//)
                GO TO   350
      306       IF(K2.EQ.0) GO TO 350
                J1=K2
                I=J
                GO TO 303
      310       IF(K3.EQ.0) GO TO 350
                J1=K3
                I=J
                GO TO 303
      314       IF(K4.EQ.0) GO TO 350
                J1=K4
*P
                I=J
                GO TO 303
      350        CONTINUE
                KNT=KK+KOUNT
                WRITE(IU,355)KNT
      355       FORMAT(10X,'TOTAL ITERATION',2X, I5)
                CLOSE(UNIT=20,FILE='IN1.DAT')
                CLOSE(UNIT=21,FILE='RES.OUT')
                STOP
                END
                SUBROUTINE MAX2(CMAX3,FMAX3,FAX2,CFAX2,M,L,II)
                DIMENSION FAX2(50),CMAX3(50,3),FMAX3(50),CFAX2(50,3),TEM
(3)
                II=3
                DO 10 I=1,M,1
                IF(FMAX3(I).GE.FMAX3(1)) GO TO 10
                TEMP=FMAX3(1)
*P
                FMAX3(1)=FMAX3(I)
                FMAX3(I)=TEMP
                DO 5 K=1,3,1
                TEM(K)=CMAX3(1,K)
                CMAX3(1,K)=CMAX3(I,K)
                CMAX3(I,K)=TEM(K)
        5       CONTINUE
       10       CONTINUE
     C
     C          STORE THE CURRENT LOWER VALUE
     C
                FAX2(L)=FMAX3(1)
                FMAX3(1)=9.1E+10
                DO 12 K=1,3,1
                CFAX2(L,K)=CMAX3(1,K)
       12       CMAX3(1,K)=9.1E+10
*
```

```
 P
                RETURN
                END
                SUBROUTINE FMAXI(B,H,FMAX1,CMAX1,L,N1,L3)
                DIMENSION B(3,50,50),H(50,50),FMAX1(50),CMAX1(50,3),TEM(
3)
                L3=3
                DO 10 I=1,N1,1
                IF(H(I,L).GE.H(1,L)) GO TO 10
                TEMP=H(1,L)
                H(1,L)=H(I,L)
                H(I,L)=TEMP
                DO 5 KK=1,L3,1
                TEM(KK)=B(KK,1,L)
                B(KK,1,L)=B(KK,I,L)
         5      B(KK,I,L)=TEM(KK)
         10     CONTINUE
        C
*P
        C       STORE THE CURRENT LOWEST VALUE AND COORDINATES
        C
                FMAX1(L)=H(1,L)
                DO 12 KK=1,L3,1
         12     CMAX1(L,KK)=B(KK,1,L)
                RETURN
                END
        C
        C       LOCAL OPTIMIZATION USING NELDER AND MEADS SMPLEX METHOD
        CC
                SUBROUTINE SMPLEX(ACL1,CACL1,CZMAX,ZMAX,DD,L,M,I4,KOUT)
                DIMENSION ACL1(50,50),CACL1(50,50,3),CZMAX(50,50,3),ZMAX
(50,50),
              1 TEMPO(50),DD(6),SUM2(3),X2(4,4),SUM(3)
                DIMENSION BST(4,4),CBST(4,4,3),CR(4,3),CP(4,3),PIMG(4,3)
,
              1 FCR(4),FPMG(4),FCW(4),FEX(4),CW(4,3)
                DIMENSION X1(4)
*P
                DIMENSION EX(4,3)
                RADD=0.00004
                I4=3
                ITER=0.
                N2=I4+1
                IU=5
                DO 3 I=1,N2,1
                BST(L,I)=+.91E+10
                DO 3 K=1,I4,1
         3      CBST(L,I,K)=(+1)**K*0.91E+11
                WRITE(IU,*)BST(L,1),BST(L,2),BST(L,3),BST(L,4)
        C
        C       FIND THE LOWEST POINT AND THE COORDINATES BST(L,1)
        C
                WRITE(IU,5) M
          5     FORMAT(10X,'THE NUMBER IS ',I4)
*
```

```
 P
            DO 10 I=1,M,1
            IF(ACL1(L,I).EQ.0.91E+10) GOTO 10
            IF(BST(L,1).LE.ACL1(L,I)) GO TO 10
            TEMP=BST(L,1)
            BST(L,1)=ACL1(L,I)
            ACL1(L,I)=TEMP
            DO 8 K=1,I4,1
            TEMPO(K)=CBST(L,1,K)
            CBST(L,1,K)=CACL1(L,I,K)
      8     CACL1(L,I,K)=TEMPO(K)
      10    CONTINUE
            WRITE(IU,*)BST(L,1)
  C
  C         FIND THE SECOND BEST
  C
            DO 20 I=1,M,1
*P
            IF(ACL1(L,I).EQ.BST(L,1)) GO TO 20
            IF(ACL1(L,I) .EQ. 0.91E+10) GOTO 20
            IF(BST(L,2).LE.ACL1(L,I))GO TO 20
            TEMP=BST(L,2)
            BST(L,2)=ACL1(L,I)
            ACL1(L,I)=TEMP
            DO 16 K=1,I4,1
            TEMPO(K)=CBST(L,2,K)
            CBST(L,2,K)=CACL1(L,I,K)
     16     CACL1(L,I,K)=TEMPO(K)

     20     CONTINUE
            WRITE(IU,*) BST(L,2)
  C
  C         FIND THE THIRD LOWEST POINT
  C
*P
            DO 26 I=1,M,1
            IF(ACL1(L,I).EQ.BST(L,1)) GO TO 26
            IF(ACL1(L,I).eq.bst(1,2)) go to 26
            IF(ACL1(L,I).EQ.0.91E+10) GOTO 26
            IF(BST(L,3).LT.ACL1(L,I)) GO TO 26
            TEMP=BST(L,3)
            BST(L,3)=ACL1(L,I)
            ACL1(L,I)=TEMP
            DO 25 K=1,I4,1
            TEMPO(K)=CBST(L,3,K)
            CBST(L,3,K)=CACL1(L,I,K)
     25     CACL1(L,I,K)=TEMPO(K)
     26     CONTINUE
            WRITE(IU,*) BST(L,3)
  c
  C         FIND THE FOURTH LOWEST POINT AT THIS TIME
*
```

```
 P
      c
            DO 29 I=1,M,1
            IF(ACL1(L,I).EQ.BST(L,1)) GO TO 29
            IF(ACL1(L,I).EQ.BST(L,2)) GO TO 29
            IF(ACL1(L,I).EQ.BST(L,3)) GO TO 29
            IF(ACL1(L,I).GE.BST(L,4)) GO TO 29
            IF(ACL1(L,I) .EQ. 0.91E+10) GOTO 29
            TEMP=BST(L,4)

            BST(L,4)=ACL1(L,I)
            ACL1(L,I)=TEMP
            DO 28 K=1,I4,1
            TEMPO(K)=CBST(L,4,K)
            CBST(L,4,K)=CACL1(L,I,K)
        28  CACL1(L,I,K)=TEMPO(K)
        29  CONTINUE
*P
            WRITE(IU,*) BST(L,4)
      C
      C  THE SIMPLEX FORMED BY BST(L,1),BST(L,2),BST(L,3)
      C  BST(L,4) WHICH

      C     IS THE BIGGEST ONE SHOULD BE REMOVED
            WRITE(IU,30) BST(L,1),BST(L,2),BST(L,3),BST(L,4)
       30   FORMAT(2X,F10.4,2X,F10.4,2X,F10.4,2X,F10.4)
      C
      C     SEE IF THE EXIT CRITERIA IS SATISFIED
      C

        31  DO 35 K=1,I4,1
            SUM(K)=0.
            DO 35 I=1,N2,1
        35  SUM(K)=SUM(K)+CBST(L,I,K)
*P
            DO 36 k=1,I4,1
        36  X1(K)=SUM(K)/4.
            DIF=0.
            DO 40 I=1,N2,1
            DO 39 K2=1,I4,1
            X2(I,K2)=CBST(L,I,K2)-X1(K2)
        39  dif=dif+x2(i,K2)**2
        40  CONTINUE
            DIFB=SQRT(DIF)
            ITER=ITER+1
            IF(DIFB.LE.RADD) GO TO 500
            WRITE(IU,43) DIFB
        43  FORMAT(4X,'DIFFERENCE= ',E12.5)
      C     REMOVE THE HIGHEST FROM THE SIMPLEX
      C     THE HIGHEST POINT IS THE BST(L,4)
      C     THE MEDIAN OF THE POINTS BST(L,1),BST(L,2),BST(L,3)
*
```

```
P
                DO 48 K=1,I4,1
                SUM2(K)=0.
                N3=N2-1
                DO 46 I=1,N3,1
          46    SUM2(K)=SUM2(K)+CBST(L,I,K)
          48    CP(L,K)=SUM2(K)/3.
      C
      C         FIND THE IMAGE OF BST(L,4) THOUGH CP
      C
                DO 50 K=1,I4,1
          50    PIMG(L,K)=2.*CP(L,K)-CBST(L,4,K)
      C
      C         CHECK IF EXCEEDS THE DOMAIN OF THE FUNCTION
      C
                DO 52 K=1,I4,1
                IF(PIMG(L,K).LT.DD(K)) PIMG(L,K)=DD(K)
*P
          52    IF(PIMG(L,K).GT.DD(K+3)) PIMG(L,K)=DD(K+3)
      C
      C         EVALUATE THE FUCTION AT THESE POINTS
      C
                L3=L
                M1=I4
                FPMG(L)=XFCT(PIMG,M1,L3)
                KOUT=KOUT+1
                write(iu,54) fpmg(l)
          54    FORMAT(4X,'FPMG=',E12.4)
                IF(FPMG(L).LT.BST(L,1)) GO TO 200
                IF(FPMG(L).LT.BST(L,2)) GO TO 100
                IF(FPMG(L).GT.BST(L,4)) GO TO 60
                DO 56 K=1,I4,1
          56    CR(L,K)=(3*CP(L,K)-CBST(L,4,K))/2.
      C
*P
      C         CHECK IF EXCEEDS DOMAIN
      C
                DO 58 K=1,K4,1
                IF(CR(L,K).LT.DD(K)) CR(L,K)=DD(K)
          58    IF(CR(L,K) .GT. DD(K+3)) CR(L,K)=DD(K+3)

      C
      C         EVALURATE THE FUNCTION
      C
                M1=I4
                L3=L
                FCR(L)=XFCT(CR,M1,L3)
                KOUT=KOUT+1
                BST(L,4)=FCR(L)
                DO 59 K=1,I4,1
          59    CBST(L,4,K)=CR(L,K)
*
```

```
P
              GO TO 400
        60    DO 65 K=1,I4,1
        65    CW(L,K)=(CP(L,K)+CBST(L,4,K))/2.
      C
      C       SEE IF EXCEEDS  DOMAIN
      C
              DO 70 K=1,I4,1
              IF(CW(L,K) .LT. DD(K)) CW(L,K)=DD(K)
         70   IF(CW(L,K).GT. DD(K+3)) CW(L,K)=DD(K+3)

              M1=I4
              L3=L
              FCW(L)=XFCT(CW,M1,L3)
              KOUT=KOUT+1
              BST(L,4)=FCW(L)
              DO 75 K=1,I4,1
*P
         75   CBST(L,4,K)=CW(L,K)
              GO  TO 400
        100   BST(L,4)=FPMG(L)
              DO 110 K=1,I4,1
        110   CBST(L,4,K)=PIMG(L,K)
              GO TO 400

        200   DO 220 K=1,I4,1
        220   EX(L,K)=3.0*CP(L,K)-2.0*CBST(L,4,K)
      C
      C       SEE IF EXCEEDS  DOMAIN
      C
              DO 250 K5=1,I4,1
              IF(EX(L,K5).LT. DD(K5))EX(L,K5)=DD(K5+3)
        250   IF(EX(L,K5).GT.DD(K5+3)) EX(L,K5)=DD(K5+3)

*P
              M1=I4
              L3=L
              FEX(L)=XFCT(EX,M1,L3)
              KOUT=KOUT+1
              IF(FEX(L).LT.BST(L,1)) GO TO  310
              WRITE(IU,255) FEX(L)
        255   FORMAT(10X,'FEX=',E10.4)
              GO TO 100
        310   BST(L,4)=FEX(L)
              DO 320 K=1,I4,1
        320   CBST(L,4,K)=EX(L,K)
        400    DO 410 I=1,N2,1
              IF(BST(L,1).LT.BST(L,I)) GO TO 410
              TEMP=BST(L,1)
              BST(L,1)=BST(L,I)
              BST(L,I)=TEMP
*
```

```
 P
            DO 405 K=1,I4,1
            TEMPO(K)=CBST(L,1,K)
            CBST(L,1,K)=CBST(L,I,K)
  405       CBST(L,I,K)=TEMPO(K)
  410       CONTINUE
            DO 450 I=2,N2,1
            IF(BST(L,2).LT.BST(L,I)) GO TO 450
            TEMP=BST(L,2)
            BST(L,2)=BST(L,I)
            BST(L,I)=TEMP
            DO 420 K=1,I4,1
            TEMPO(K)=CBST(L,2,K)
            CBST(L,2,K)=CBST(L,I,K)
  420       CBST(L,I,K)=TEMPO(K)
  450       CONTINUE
            IF(BST(L,3).LT.BST(L,4)) GO TO 460
*P
            TEMP=BST(L,3)
            BST(L,3)=BST(L,4)
            BST(L,4)=TEMP
            DO 455 K=1,I4,1
            TEMPO(K)=CBST(L,3,K)
            CBST(L,3,K)=CBST(L,4,K)
  455       CBST(L,4,K)=TEMPO(K)
  460       IF(ITER.LT.10.)GOTO 31
            ITER=0.
            BST(L,4)=(BST(L,1)+BST(L,2)+BST(L,3)+BST(L,4))/4.
            GO TO 400
  500       ZMAX(L,1)=BST(L,1)
            DO 505 K=1,I4,1
  505       CZMAX(L,1,K)=CBST(L,1,K)
            WRITE(IU,510) ZMAX(L,1),(CZMAX(L,1,K),K=1,I4)
  510       FORMAT(10X,E10.4,3(E10.4,4X),/)
*P
            RETURN
            END
            FUNCTION XFCT(C,II,IP)
            DIMENSION C(4,3)
            II=3
            XFCT=9.0-8.0*C(IP,1)-6.0*C(IP,2)-4.0*C(I
         1  P,3)+2.0*C(IP,1)**2+2.0*C(IP,2)**2+C(IP,3)**2
         2  +C(IP,1)*C(IP,2)*2.0+2.0*C(IP,1)*C(IP,3)
            RETURN
            END
            FUNCTION CAL(C1,L3,N3)
            DIMENSION C1(3,50,50)
            CAL=9.0-8.0*C1(1,L3,N3)-6.0*C1(2,L3,N3)-4.0*C1(3,L3,N3)+
         1  2.0*C1(1,L3,N3)**2+2.0*C1(2,L3,N3)**2+C1(3,L3,N3)**2
         2  +2.0*C1(1,L3,N3)*C1(2,L3,N3)+2.0*C1(1,L3,N3)*C1(3,L3,N3)
            RETURN
*
```